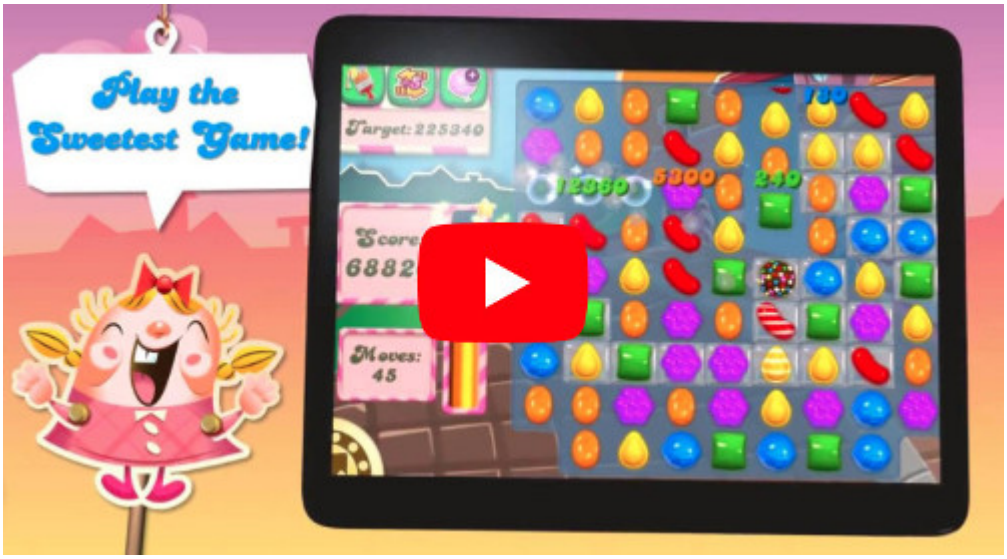
1. Candy Crush Saga

Candy Crush Saga (https://king.com/game/candycrush) is a hit mobile game developed by King (part of Activision|Blizzard) that is played by millions of people all around the world. The game is structured as a series of levels where players need to match similar candy together to (hopefully) clear the level and keep progressing on the level map. If you are one of the few that haven't played Candy Crush, here's a short demo:



(https://youtu.be/HGLGxnfs_t8)

Candy Crush has more than 3000 levels, and new ones are added every week. That is a lot of levels! And with that many levels, it's important to get *level difficulty* just right. Too easy and the game gets boring, too hard and players become frustrated and quit playing.

In this project, we will see how we can use data collected from players to estimate level difficulty. Let's start by loading in the packages we're going to need.

In [184]:
```
# This sets the size of plots to a good default.
options(repr.plot.width = 5, repr.plot.height = 4)
library(readr)
library(dplyr)
library(ggplot2)
```

2. The data set

The dataset we will use contains one week of data from a sample of players who played Candy Crush back in 2014. The data is also from a single *episode*, that is, a set of 15 levels. It has the following columns:

- **player_id**: a unique player id
- **dt**: the date
- **level**: the level number within the episode, from 1 to 15.
- **num_attempts**: number of level attempts for the player on that level and date.
- **num_success**: number of level attempts that resulted in a success/win for the player on that level and date.

The granularity of the dataset is player, date, and level. That is, there is a row for every player, day, and level recording the total number of attempts and how many of those resulted in a win.

In [186]:
```
# Reading in the data
data <- read_csv("datasets/candy_crush.csv")

head(data)
```

```
Parsed with column specification:
cols(
  player_id = col_character(),
  dt = col_date(format = ""),
  level = col_double(),
  num_attempts = col_double(),
  num_success = col_double()
)
```

A tibble: 6 x 5

player_id	dt	level	num_attempts	num_success
<chr>	<date>	<dbl>	<dbl>	<dbl>
6dd5af4c7228fa353d505767143f5815	2014-01-04	4	3	1
c7ec97c39349ab7e4d39b4f74062ec13	2014-01-01	8	4	1
c7ec97c39349ab7e4d39b4f74062ec13	2014-01-05	12	6	0
a32c5e9700ed356dc8dd5bb3230c5227	2014-01-03	11	1	1
a32c5e9700ed356dc8dd5bb3230c5227	2014-01-07	15	6	0
b94d403ac4edf639442f93eeffdc7d92	2014-01-01	8	8	1

3. Checking the data set

Now that we have loaded the dataset let's count how many players we have in the sample and how many days worth of data we have.

In [188]:
```
# Count and display the number of unique players
print("Number of players:")
length(unique(data$player_id))

# Display the date range of the data
print("Period for which we have data:")
range(data$dt)
```

```
[1] "Number of players:"
```

6814

```
[1] "Period for which we have data:"
```

2014-01-01 2014-01-07

4. Computing level difficulty

Within each Candy Crush episode, there is a mix of easier and tougher levels. Luck and individual skill make the number of attempts required to pass a level different from player to player. The assumption is that difficult levels require more attempts on average than easier ones. That is, *the harder* a level is, *the lower* the probability to pass that level in a single attempt is.

A simple approach to model this probability is as a Bernoulli process (https://en.wikipedia.org/wiki/Bernoulli_process); as a binary outcome (you either win or lose) characterized by a single parameter p_{win}: the probability of winning the level in a single attempt. This probability can be estimated for each level as:

$$p_{win} = \frac{\sum wins}{\sum attempts}$$

For example, let's say a level has been played 10 times and 2 of those attempts ended up in a victory. Then the probability of winning in a single attempt would be p_{win} = 2 / 10 = 20%.

Now, let's compute the difficulty p_{win} separately for each of the 15 levels.

In [190]:
```
# Calculating level difficulty
difficulty <- data%>%
 group_by(level)%>%
 summarise(attempts = sum(num_attempts), wins = sum(num_success)) %>%
 mutate(p_win = wins/attempts)

head(difficulty)
```

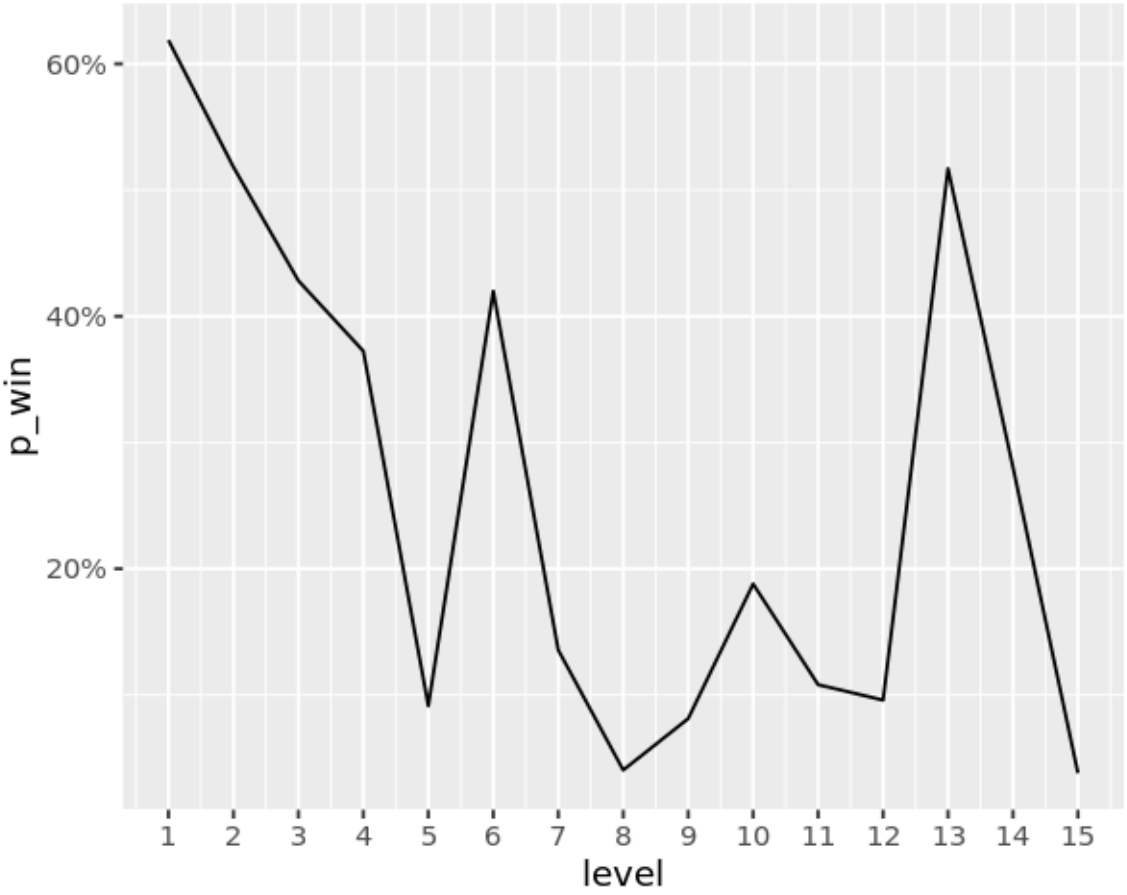
A tibble: 6 x 4

level	attempts	wins	p_win
<dbl>	<dbl>	<dbl>	<dbl>
1	1322	818	0.61875946
2	1285	666	0.51828794
3	1546	662	0.42820181
4	1893	705	0.37242472
5	6937	634	0.09139397
6	1591	668	0.41986172

5. Plotting difficulty profile

Great! We now have the difficulty for all the 15 levels in the episode. Keep in mind that, as we measure difficulty as the probability to pass a level in a single attempt, a *lower* value (a smaller probability of winning the level) implies a *higher* level difficulty.

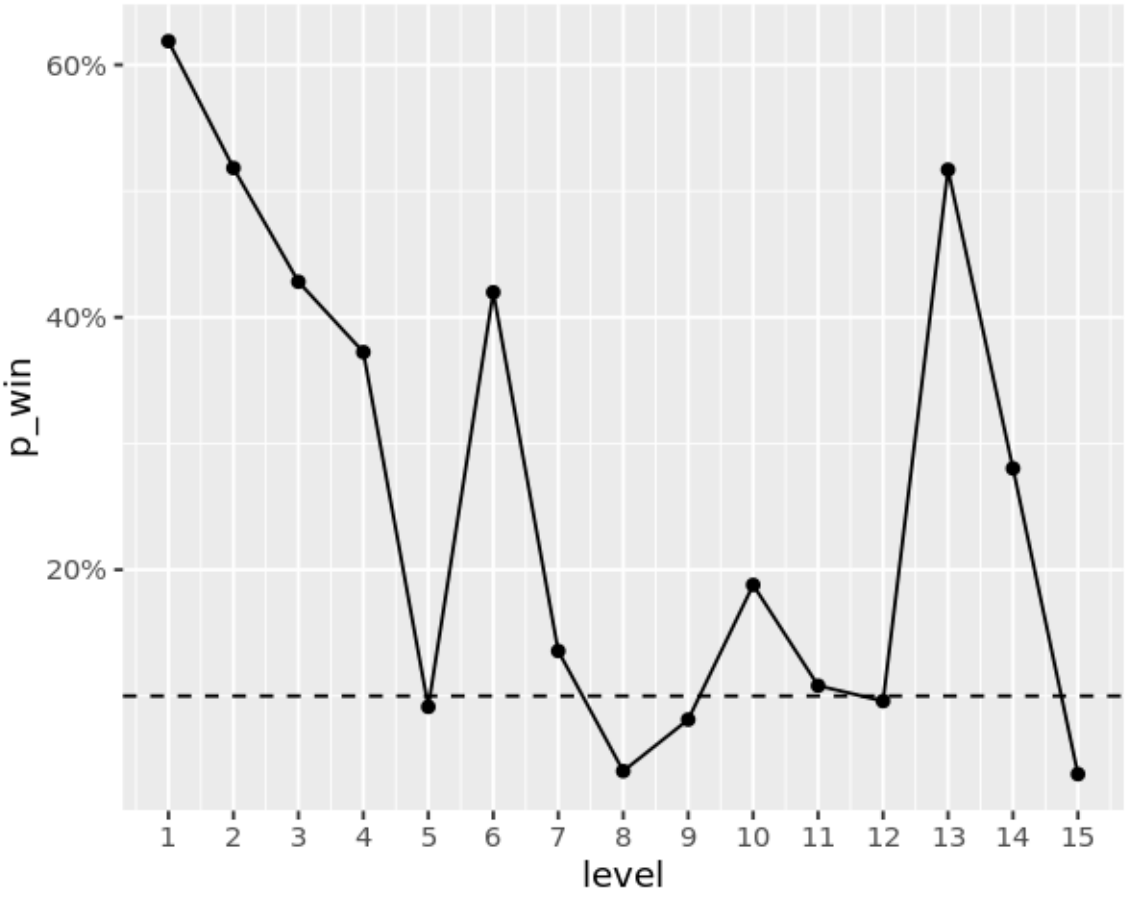
In [192]:
```
# Plotting the level difficulty profile
ggplot(difficulty, aes(level, p_win)) +  geom_line() +
        scale_x_continuous(breaks = 1:15) +
        scale_y_continuous(label = scales::percent)
```



6. Spotting hard levels

What constitutes a *hard* level is subjective. However, to keep things simple, we could define a threshold of difficulty, say 10%, and label levels with p_{win} < 10% as *hard*. It's relatively easy to spot these hard levels on the plot, but we can make the plot more friendly by explicitly highlighting the hard levels.

In [194]:
```
# Adding points and a dashed line
ggplot(difficulty, aes(level, p_win)) +  geom_line() +
        geom_point() +
        geom_hline(yintercept = 0.1, linetype = "dashed") +
        scale_x_continuous(breaks = 1:15) +
        scale_y_continuous(label = scales::percent)
```



7. Computing uncertainty

As Data Scientists we should always report some measure of the uncertainty of any provided numbers. Maybe tomorrow, another sample will give us slightly different values for the difficulties? Here we will simply use the *Standard error* (https://en.wikipedia.org/wiki/Standard_error) as a measure of uncertainty:

$$\sigma_{error} \approx \frac{\sigma_{sample}}{\sqrt{n}}$$

Here n is the number of datapoints and σ_{sample} is the sample standard deviation. For a Bernoulli process, the sample standard deviation is:

$$\sigma_{sample} = \sqrt{p_{win}(1 - p_{win})}$$

Therefore, we can calculate the standard error like this:

$$\sigma_{error} \approx \sqrt{\frac{p_{win}(1 - p_{win})}{n}}$$

We already have all we need in the `difficulty` data frame! Every level has been played *n* number of times and we have their difficulty p_{win}. Now, let's calculate the standard error for

In [196]:
```
# Computing the standard error of p_win for each level
difficulty <- difficulty %>%
    group_by(level)%>%
    mutate(error = sqrt(p_win*(1-p_win)/attempts))

head(difficulty)
```

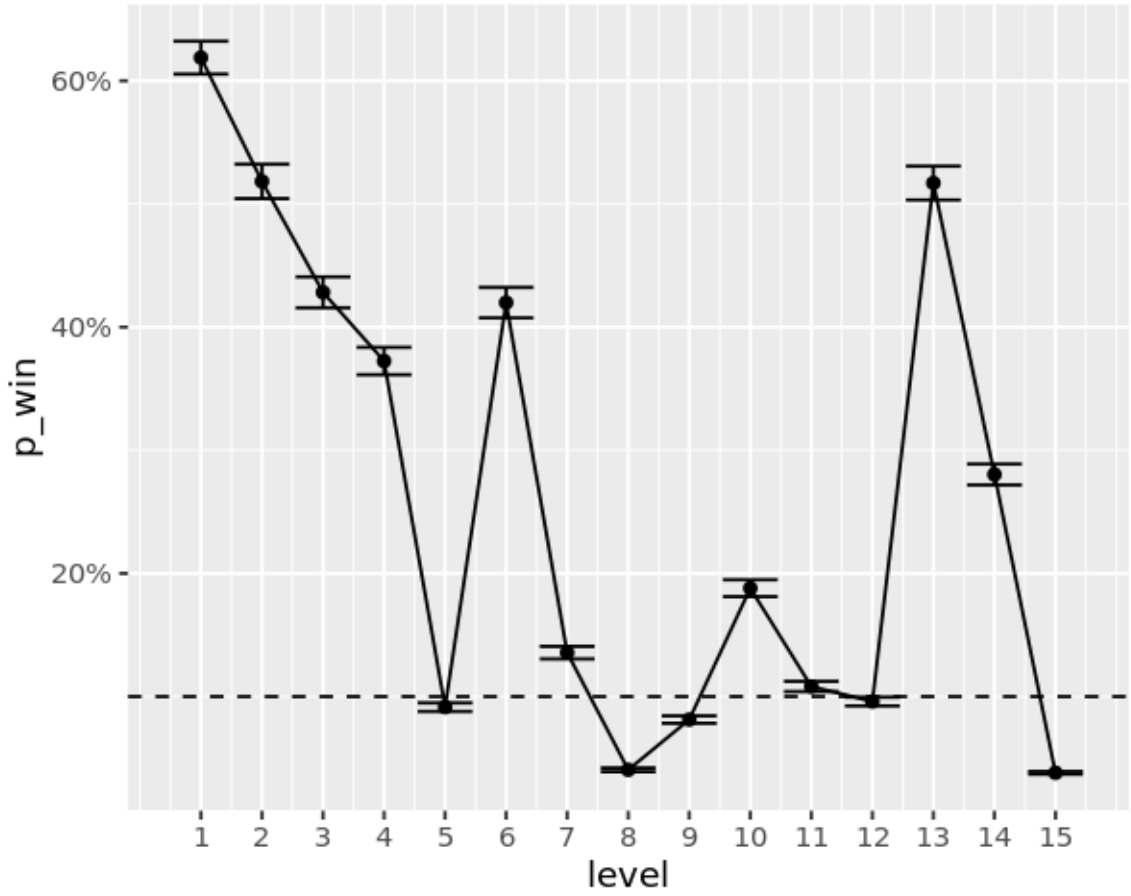
A grouped_df: 6 x 5

level	attempts	wins	p_win	error
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1322	818	0.61875946	0.013358101
2	1285	666	0.51828794	0.013938876
3	1546	662	0.42820181	0.012584643
4	1893	705	0.37242472	0.011111607
5	6937	634	0.09139397	0.003459878
6	1591	668	0.41986172	0.012373251

8. Showing uncertainty

Now that we have a measure of uncertainty for each levels' difficulty estimate let's use *error bars* to show this uncertainty in the plot. We will set the length of the error bars to one standard error. The upper limit and the lower limit of each error bar should then be p_{win} + σ_{error} and p_{win} - σ_{error}, respectively.

In [198]:
```
# Adding standard error bars
ggplot(difficulty, aes(level, p_win)) +  geom_line() +
        geom_point() +
        geom_hline(yintercept = 0.1, linetype = "dashed") +
        scale_x_continuous(breaks = 1:15) +
        scale_y_continuous(label = scales::percent)+
        geom_errorbar(aes(ymin = p_win - error, ymax = p_win + error))
```



9. A final metric

It looks like our difficulty estimates are pretty precise! Using this plot, a level designer can quickly spot where the hard levels are and also see if there seems to be too many hard levels in the episode.

One question a level designer might ask is: "How likely is it that a player will complete the episode without losing a single time?" Let's calculate this using the estimated level difficulties!

In [200]:
```
# The probability of completing the episode without losing a single time
p <- prod(difficulty$p_win)

# Printing it out
p
```

9.44714093448606e-12

10. Should our level designer worry?

Given the probability we just calculated, should our level designer worry about that a lot of players might complete the episode in one attempt?

In [202]:
```
# Should our level designer worry about that a lot of
# players will complete the episode in one attempt?
should_the_designer_worry = FALSE
```